NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
FINAL ASSESSMENT FOR
Semester 1 AY2018/2019

CS1010 Programming Methodology

November 2018 Time Allowed 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains 15 questions and comprises 14 printed pages, including this page.
2. Write all your answers in the answer sheet provided.
3. The total marks for this assessment is 70. Answer **ALL** questions.
4. This is an **OPEN BOOK** assessment.
5. You can assume that in all the code given, no overflow nor underflow will occur during execution. In addition, you can assume that all given inputs are valid and fits within the specified variable type.
6. You can assume all the necessary headers (`math.h`, `stdbool.h`, `cs1010.h`, etc.) are included. For brevity, the include directives are not shown. Further, not all variable declarations are shown. You can assume that all variables are properly declared with the right type.
7. State any additional assumption that you make.
8. Please write your student number only. Do not write your name.

Part I

Multiple Choice Questions (36 points)

For each of the questions below, **write your answer in the corresponding answer box on the answer sheet.** Each question is worth 3 points.

If multiple answers are equally appropriate, pick one and write the chosen answer in the answer box. Do NOT write more than one answer in the answer box.

1. (3 points) Consider the following three functions that take in a `long` value as input. Which of these three functions are equivalent? Two functions are equivalent if both return the same value when given the same input.

```
bool f(long x) {
  if (x % 10 == 0) {
    if (x < 0) {
      return true;
    }
  } else {
    if (x < 10) {
      return true;
    }
  }
  return false;
}

bool g(long x) {
  if (x > 10 && x % 10 > 0) {
    return false;
  }
  return true;
}

bool h(long x) {
  return (x < 0 && x % 10 == 0) || (x < 10);
}
```

   A. `f` and `g` only
   B. `g` and `h` only
   C. `g` and `h` only
   D. `f`, `g`, and `h`

   Write `X` in the answer box if no two functions are equivalent.

2. (3 points) Consider the following three functions that takes in an array with at least one element as input. Which of these three functions are equivalent? Two functions are equivalent if they always return the same value when given the same input.

```
long foo(long len, long a[len]) {
  for (long i = 0; i < len; i += 1) {
    if (a[i] % 2 == 0) {
      return i;
    }
  }
  return len;
}

long bar(long len, long a[len]) {
  long i = 0;
  bool done = false;
  while (!done) {
    if (a[i] % 2 == 0 || i >= len) {
      done = true;
    }
    i += 1;
  }
  return i;
}

long qux(long len, long a[len]) {
  long i = -1;
  do {
    i += 1;
  } while (a[i] % 2 != 0 && i < len);
  return i;
}
```

A. foo and bar only

B. foo and qux only

C. bar and qux only

D. foo, bar, and qux

Write X in the answer box if no two functions are equivalent.

3. (3 points) Suppose we allocate a dynamic array as follows:

```
char *str = calloc(len, sizeof(char));
```

Which of the following snippet would lead to illegal access of memory?

```
void moo(char *str, long len) {
  for (long i = 0; i <= len; i++) {
    str[i] = 'a';
  }
  cs1010_println_string(str);
}

void baa(char *str, long len) {
  for (long i = 0; i < len; i++) {
    str[i] = 'a';
  }
  cs1010_println_string(str);
}

void quack(char *str, long len) {
  for (long i = 1; i < len-1; i++) {
    str[i] = 'a';
  }
  cs1010_println_string(str);
}
```

A. moo and baa only

B. moo and quack only

C. moo, baa, and quack

D. moo only

E. quack only

Write X in the answer box if none of the combinations above is correct.

4. (3 points) Consider the code below:

```
double d;
while (d > 0) {
  d -= 1;
}
```

Which of the following statements about the code above is TRUE?

A. The code causes a compilation error because `-=` can only be used with integer types

B. The code causes a compilation error because `d` is not initialized

C. The code causes a compilation error because we should cast `0` and `1` to `double`

D. The code may lead to an infinite loop because `d` is not initialized

E. The code may lead to an infinite loop because `0` cannot be precisely represented by the type `double`

Write `X` in the answer box if none of the statements above is true.

5. (3 points) Suppose we define a macro AVERAGE as follows:

```
#define AVERAGE(x,y) (0.5 * x) + (0.5 * y)
```

What is the value of `double_average`, round to the nearest integer, after executing the three lines of code below?

```
long x = 10;
long y = 20;
double double_average = 2*AVERAGE(x, y);
```

A. 20

B. 25

C. 30

D. 35

E. 40

Write `X` in the answer box if none of the answers above is correct.

6. (3 points) Consider the code below:

```
void tata(double *ptr, double x) {
  ptr = &x;
}

void titi(double *ptr, double *x) {
  *ptr = *x;
}

void tete(double **ptr, double x) {
  *ptr = &x;
}

void tutu(double **ptr, double *x) {
  *ptr = x;
}

int main()
{
  double *ptr;
  double x;
  double y;
  ptr = &y;
  // Line B
}
```

Which of the following function invocation at Line B would cause the assertion `ptr == &x` to be true?

A. `tata(ptr, x);`

B. `titi(ptr, &x);`

C. `tete(&ptr, x);`

D. `tutu(&ptr, &x);`

E. `tutu(*ptr, *x);`

Write X in the answer box if none of the answers above is correct.

7. (3 points) Which of the following functions run in $O(n^2)$ time?

```
void egg(long n, long a[n]) {
  for (long i = 0; i < n/2; i += 2) {
    for (long j = i/2; j < n; j += 1) {
      cs1010_println_long(i);
      cs1010_println_long(j);
    }
  }
}

void ham(long n, long a[n]) {
  for (long i = 1; i < pow(2, n); i *= 2) {
    for (long j = 1; j < n; j += 1) {
      cs1010_println_long(i);
      cs1010_println_long(j);
    }
  }
}

void cheese(long n, long a[n]) {
  for (long i = 1; i < n*sqrt(n); i += 1) {
    for (long j = 1; j < sqrt(n); j += 1) {
      cs1010_println_long(i);
      cs1010_println_long(j);
    }
  }
}
```

A. egg only

B. cheese only

C. ham and cheese only

D. egg and ham only

E. egg, ham, and cheese

Write X in the answer box if none of the combinations above is correct.

8. (3 points) Consider the code below:

```
bool mystery(long a[], long start, long end) {
  if (end > start) {
    return false;
  }
  long mid = (start + end)/2;
  long sum = 0;
  for (long i = start; i <= mid; i += 1) {
    sum += a[i];
  }
  if (sum == 0) {
    return true;
  }
  if (sum > 0) {
    return mystery(a, start, mid);
  }
  return mystery(a, mid+1, end);
}
```

What is the worst-case running time of `mystery`, expressed using Big-O notation, when the input is an array of size n?

A. $O(n^2)$

B. $O(n \log n)$

C. $O(n)$

D. $O(\log^2 n)$

E. $O(\log n)$

Write X in the answer box if none of the answers above is correct.

Questions 9 to 10 are based on the following function.

```
void do_something(long len, long a[len]) {
  long curr = 0;
  while (curr < len) {
    if (curr == 0 || a[curr] >= a[curr-1]) {
      curr += 1;
    } else {
      long temp = a[curr];
      a[curr] = a[curr - 1];
      a[curr - 1] = temp;
      curr -= 1;
      // Line A
    }
  }
}
```

9. (3 points) Suppose we have an array `long a[3] = {3, 1, 2};`. What is the content of the array `a` after calling `do_something(3, a);` ?

   A. `{3, 2, 1}`

   B. `{1, 2, 3}`

   C. `{3, 3, 3}`

   D. `{3, 1, 2}`

   E. `{2, 1, 3}`

   Write X in the answer box if none of the answers above is correct.

10. (3 points) Which of the following is a correct assertion at Line A of the `do_something`?

    A. `{ curr != 0 && a[curr-1] > a[curr-2] }`

    B. `{ curr != 0 && a[curr] > a[curr-1] }`

    C. `{ curr != 0 && a[curr + 1] > a[curr] }`

    D. `{ curr >= 0 && a[curr] > a[curr-1] }`

    E. `{ curr >= 0 && a[curr + 1] > a[curr] }`

    Write X in the answer box if none of the answers above is correct.

11. (3 points) Trace through the code below:

```
#include "cs1010.h"

struct obj {
  long *ptr;
  long id;
};

void blah(struct obj *optr) {
  optr->id = 20;
  optr->ptr = &optr->id;
}

int main()
{
  struct obj o;
  long x = 10;

  o.ptr = &x;
  o.id = 0;

  blah(&o);

  cs1010_println_long(o.id);
  cs1010_println_long(*(o.ptr));

}
```

Which two numbers will be printed by the program above?

A. `20` and `10`

B. `0` and `10`

C. `0` and `0`

D. `10` and `10`

E. `20` and `20`

Write `X` in the answer box if none of the answers above is correct.

12. (3 points) Consider the following implementation of insertion sort:

```
void insert(long a[], long curr)
{
  long i = curr - 1;
  long temp = a[curr];
  while (temp <= a[i] && i >= 0) {
    a[i+1] = a[i];
    i -= 1;
  }
  a[i+1] = temp;
}

void insertion_sort(long n, long a[n]) {
  for (long curr = 1; curr < n; curr += 1) {
    insert(a, curr);
  }
}
```

Which of the following statement(s) is/are true about the code above:

(i) If the input array a contains n elements, all has the same value, `insertion_sort` takes $O(n^2)$ time.

(ii) If the input array a contains n distinct elements that are sorted in descending order, `insertion_sort` takes $O(n)$ time.

(iii) If the input array a contains n distinct elements that are sorted in ascending order, `insertion_sort` takes $O(1)$ time.

A. Only (i)

B. Only (i) and (ii)

C. Only (i) and (iii)

D. Only (ii) and (iii)

E. (i), (ii), and (iii)

Write X in the answer box if none of the combinations is correct.

Part II

Short Questions (34 points)

Answer all questions in the space provided on the answer sheet. Be succinct and write neatly.

13. (6 points) **Binary.** The following program `binary` generates all binary strings (i.e., strings consisting of `'0'` and `'1'` only) of a given length `n` recursively. For instance

```
ooiwt@pe101:~$ ./binary
2
00
01
10
11
ooiwt@pe101:~$ ./binary
3
000
001
010
011
100
101
110
111
```

The function `generate` below recursively generates all binary substrings of length `n - k` and prints out the binary string. The recursive calls, however, are missing.

Complete the function `generate`. Write only the missing lines on the answer sheet.

```
void generate(long n, char str[], long k) {
  if (k == n-1) {
    str[k] = '0';
    cs1010_println_string(str);
    str[k] = '1';
    cs1010_println_string(str);
    return;
  }

  // Missing Lines

}
```

14. (15 points) **Search.** Suppose we have the following function

`long binsearch(const long list[], long i, long j, long q)`

that looks for the item q in an array `list`, among items `list[i]` .. `list[j]` using binary search. The code for this function is omitted.

In an attempt to try to speed up binary search on a large array, Mario wrote the following function to find the starting point and end point in the array that contains q.

```
long narrowing_then_search(const long list[], long len, long q) {
  long start = 0;
  long end = 1;
  do {
    if (q == list[end]) {
      return end;
    }
    if (q < list[end]) {
      // Line F
      return binsearch(list, start, end, q);
    }
    start = end;
    end += 10; // Line G
    // Line H
  } while (end < len);
  // Line I
  return binsearch(list, start, len-1, q);
}
```

You can assume that the input array is already sorted, in non-decreasing order.

(a) (3 points) Write an assertion that relates q to `list[start]` in Line H.

(b) (3 points) Write an assertion that relates q to `list[start]` and `list[end]` in Line F.

(c) (3 points) Write an assertion that relates q to `list[start]` and `list[len-1]` in Line I.

(d) (3 points) What is the worst case running time, in Big-O notation, of this algorithm?

(e) (3 points) Suppose we change Line G to

`end *= 2;`

What is the worst-case running time, in Big-O notation, now?

15. (13 points) **Stack.**

Consider the code below:

```
void f(__ a, __ b) {
  *(b + 1) = a;
  b = &a;
  // Line D
}

int main() {
  long x[2] = {-5, 10};
  long *p;

  p = x;
  f(*p, x);

  // Line E
}
```

(a) (4 points) The types for the parameters `a` and `b` to function `f` is missing. Fill in the correct type of the parameters so that the compiler does not report any error or warning.

(b) (7 points) Draw the content of the call stack when the execution reaches Line D using the notations similar to what has been used in CS1010. Label all your call frames, variables, and values on the call stack. You may use arrows to denote pointers, instead of using the actual memory addresses.

(c) (2 points) What are the values in the array `x` after calling `f`, at Line E?

END OF PAPER